Object-Oriented Control System Design Using On-Line Training of Artificial Neural Networks

Final Report
Grant No. NAG3-1661

December 01, 1996 - April 30, 1997
(FINAL REPORT)

Howard University/NASA Lewis Cooperative
Research Studies

Mr. Donald Noga
Technical Officer

Ahmed Rubaai, Dr. Eng.
Principal Investigator
Associate Professor of Electrical Engineering

Howard University
College of Engineering, Architecture and Computer Sciences
Electrical Engineering Department
2300 6th Street, Northwest
Washington, DC 20059
E-mail: rubaai@scs.howard.edu

TABLE OF CONTENTS

Object-Oriented Control System Design Using On-Line Training of Artificial Neural Networks

Ahmed Rubaai, Member, IEEE
Electrical Engineering Department
Howard University
Washington, DC 20059
E-mail: rubaai@scs.howard.edu

ABSTRACT

This report deals with the object-oriented model development of a neuro-controller design for permanent magnet (PM) dc motor drives. The system under study is described as a collection of interacting objects. Each object module describes the object behaviors, called methods. The characteristics of the object are included in its variables. The knowledge of the object exists within its variables, and the performance is determined by its methods. This structure maps well to the real world objects that comprise the system being modeled. A dynamic learning architecture that possesses the capabilities of simultaneous on-line identification and control is incorporated to enforce constraints on connections and control the dynamics of the motor. The control action is implemented "on-line", in "real time" in such a way that the predicted trajectory follows a specified reference model. A design example of controlling a PM dc motor drive on-line shows the effectiveness of the design tool. This will therefore be very useful in aerospace applications. It is expected to provide an innovative and noval software model for the rocket engine numerical simulator executive.

I. INTRODUCTION

Complex systems such as airplanes, automobiles, rocket engine, and so on are often composed of hundreds or even thousands of objects. To ensure accurate modeling, an enormous number of state variables and behaviors for system components must be controlled. Because all necessary controlling features are built in, instead of being accessed through call statements, the model is compact and readable, making it much easier to develop, maintain and enhance.

Object-oriented model development is currently a popular software paradigm that is gaining wide acceptance because it allows us to build rapid prototypes of system model and add details as our knowledge of the system models under study increases [1-3]. The application of object oriented programming to power system simulation, analysis, education and control have been explored in this field [4-8]. The modeling techniques presented drew heavily on object-oriented technology while leaving open the degree to which object-oriented languages and/or data-bases would be employed in actual implementation. The feasibility of using physical objects as the basis for distributed message passing on-line network applications was also demonstrated. This work indicates that all network analysis programs may be designed with object orientation.

Motor drives are complex control objects. There are risks in replacing conventional control with adaptive control as this requires identification of an on-line linear model. To ensure good performance of adaptive control, several parameters must be chosen. A design tool is necessary to adapt to the uncertainities of the motor dynamics and in addition to learn about its inherent nonlinearities.

In this report, an object-oriented motor drive control design tool is proposed. It integrates modelling, learning, design, analysis and graphics user interface in one package. This facility can be conveniently used to study motor dynamic behavior and develop modern control algorithms. The control system design tool is a comprehensive package integrating neural control system design and learning processes. It is a flexible design environment capable of learning the unknown nonlinear dynamics of different motors, analyzing and designigning different control strategies. These include noise rejection, uncertainities of motor dynamics and closed-loop systems.

The neural control system design consists of a 3 layer Feed-forward Artificial Neural-Network (FANN) with hidden layers. Each hidden layer is capable of housing an arbitrary number of neurons. It is possible to specify an arbitrary sigmoid activation function and its derivatives for the neurons in any given hidden layer. The control signal is generated in such a way that the trajectory follows a specified reference model.

II. STRUCTURE

The object-oriented motor drive control system design tool, as shown in figure 1, is comprised mainly of four parts: motor modelling, dynamic learning, neural control strategy and controller optimization. The design is based on motor types and knowledge base including object models, a control library and a database. Each motor or controller component is treated as an object which consists of data and a source code. Complete system models can be built by connecting objects, as in motor drives.

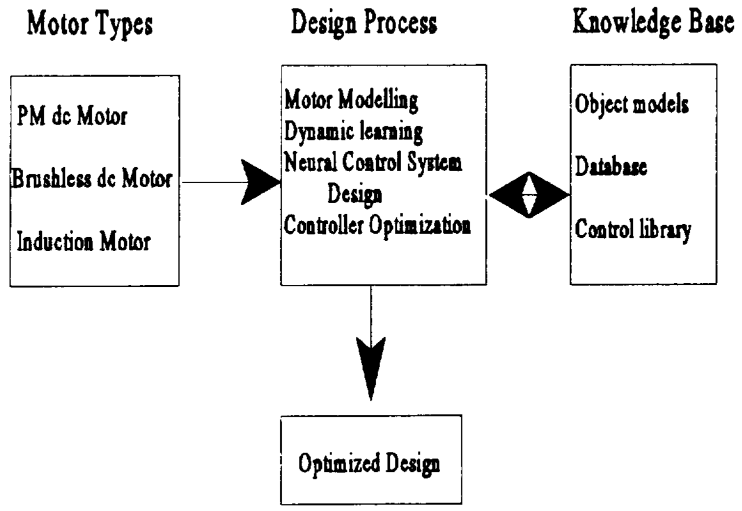

Fig. 1. Design Structure

III. MODELLING

A motor drive system is a combination of related components such as motor, high speed switching converter and control systems. Using the design tool, control strategies and performance studies become extremely convenient. Each component of both motor and control systems is treated as an object, and a complete system model can be built by connecting objects. If several objects share the same data or equations (methods), these can be stored in class, rather than creating a separate object for each. This class is saved in a knowledge base. The core of the knowledge base consists of mathematical models of objects.

A complete model of a PM dc motor and its control system was constructed using the design tool. The schematic diagram of the model is shown in figure 2. The structure of the model is based on the following: 1) the choice of "C" as the implementation language, 2) the use of object-oriented techniques in the resultant implementation, and 3) the use of Dynamic Back-Propagation (DBP) learning algorithm to train the artificial neural network on-line. The following is a brief description of the main components of the developed model:

- Subsystem "A" encloses a system for calculating the desired motor speed at any time instant. This function is performed for each reference speed track.

- Subsystem "B" encloses a system for : 1) calculates the instantaneous terminal voltage that forces the motor to follow the reference model. This function takes the 3 by 1 matrix that contains the motor speeds at 3 successive time instants and returns the desired voltage for the motor to track the reference model, and 2) uses the instantaneous terminal voltage, and the motor speeds at the past two instants to compute the actual motor speed. This function takes the 2 by 1 matrix of the motor speeds at 2 successive time instants, the network approximation to the terminal voltage, and returns the actual speeds at the next instant.

- Subsystem "C" encloses a system that takes the input vector to the network and computes: 1) the input to all hidden layers. This function takes the output matrix, of the previous hidden layer, the weight and bias matrices that are between the previous and the current (receiver) hidden layer and returns the input matrix to the current hidden layer, 2) the derivatives of the input to all hidden layers. This function takes the matrix , of inputs to the current hidden layer and computes their derivatives, and 3) the output of all hidden layers. This function takes the matrix of inputs to the current hidden layer and computes the output matrix

- Subsystem "D" encloses a system that contains the systems with Subsystem "C", and: 1) computes the partial derivatives of the network output with respect to its parameters. This function returns network output to a given input matrix and also computes the partial derivatives of each network output with respect to its adjustable parameters, and 2) updates the weights and biases of the network using the DBP algorithm.This function takes the network parameters, and the error matrix as its input. It updates the weight and bias matrices according to the DBP algorithm.

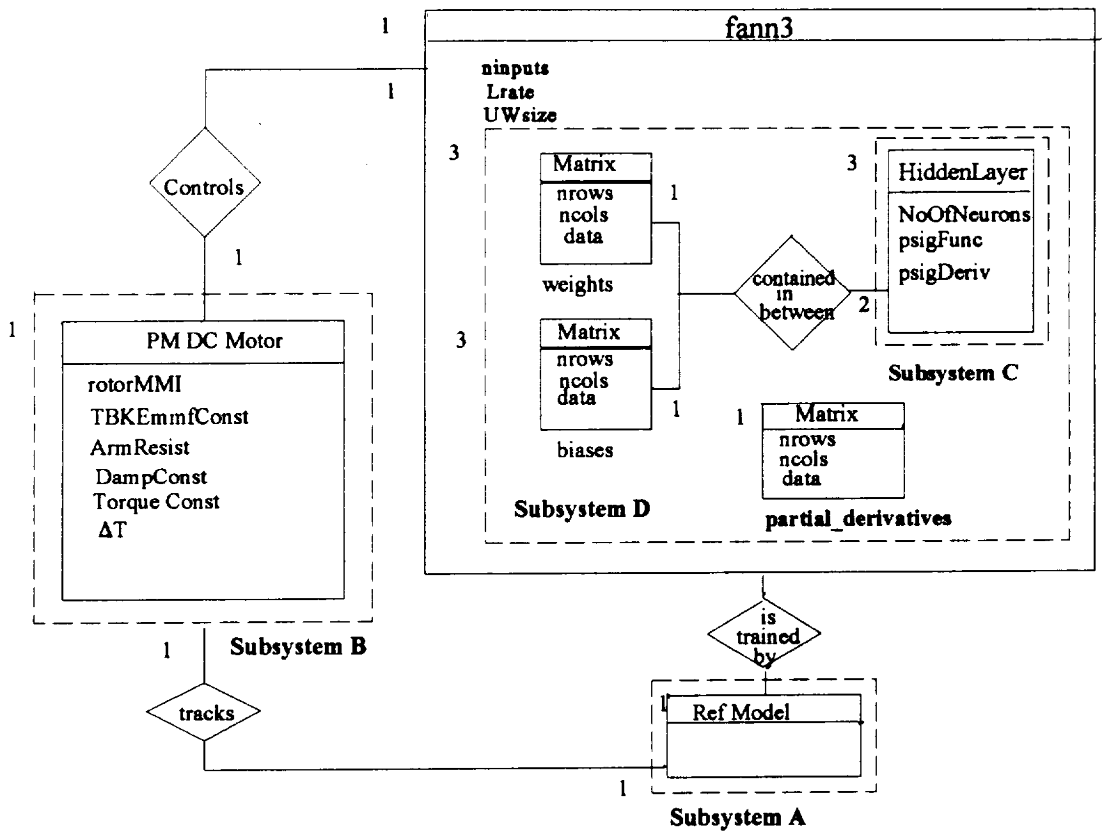

Figure 2. Object Model for the Motor Drive Control Structure

IV. DYNAMIC LEARNING

The first stage in learning is to choose the neural configuration. The neural network considered here is a feedforward network with sigmoid functions. This class of neural networks is well known and easily implemented under real-time conditions. The network configuration (the number of hidden layers, and the number of neurons in each hidden layer) was chosen heuristically on a trial and error basis. Neither the number of hidden layers nor the number of neurons in each layer are known a-priori. However the number of neurons in the input and output layers of the FANN are fixed by the PM dc motor dynamics and the discretization scheme used in this report. The Dynamic Back-Propagation (DBP) training algorithm is used to perform the real-time identification and control of the PM dc motor drive. The DBP algorithm is a straight-forward extension of the conventional (static) Error Back-Propagation Training Algorithm, applied to dynamic systems. It was first introduced by Narendra and Parthasarathy, [9], and is based on the principle of the minimization of a cost function of the error between the desired output and the actual output of a Feed-forward Artificial Neural Network (FANN). The minimization is achieved by varying the adjustable parameters of the FANN in the direction of the gradient of the cost function. Figure 3, shows the architecture of a 3 layer FANN with n_0 neurons in the input layer, and n_i neurons in each of its ith hidden layers. The FANN is a non-linear transformation of $u(k) \in \mathbb{R}^{n0}$ to $y(k) \in \mathbb{R}^{n3}$, where $u(k) = [u_1(k)\ u_2(k)\ \ldots\ u_{n0}(k)]^T$, $y(k) = [y_1(k)\ y_2(k)\ \ldots\ y_{n3}(k)]^T$ and $y_d(k) = [y_{d1}(k)\ y_{d2}(k)\ \ldots\ y_{dn3}(k)]^T$ is the desired output vector at time $t = k\Delta T$. Each neuron in the ith hidden layer of the network consists of a nonlinear mapping, that is usually chosen to be a sigmoidal function of the form $\gamma_i(x) = (1 - \exp(-x))/(1 + \exp(-x))$.

where, W_1, b_1, W_2, b_2, and W_3, b_3 are the weight matrices and the bias vectors of the network. In the DBP algorithm, it is useful to rearrange the elements of the weight matrices and bias vectors into a vector, θ, of adjustable parameters of the network. The cost function in the DBP algorithm is chosen to be:

$$J(\theta) = \frac{1}{T} \sum_{n=k-T+1}^{k} \sum_{j=1}^{n_3} [y_j(n) - y_{dj}(n)]^2 \qquad (1)$$

The parameter T, is referred to as the update window size, and equals the number of time instants over which the gradient of J is computed. The DBP algorithm begins by initially assigning small randomly chosen values for the weights and biases. The training is terminated when, $\|y(k) - y_d(k)\|$ falls below a user-specified tolerance, for some k. The most important step in the DBP algorithm is the computation of the partial derivatives of each of

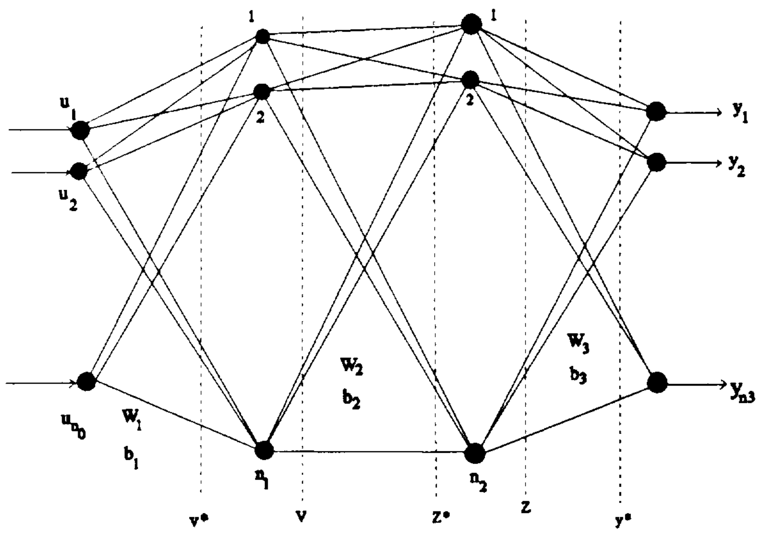

Fig. 3 Architecture of a 3 Layer FANN

the outputs of the network with respect to each of its adjustable parameters. These partial derivatives are used in computing the gradient of J, every T instants. Details of the computation of the partial derivatives are not addressed in this paper and are given in earlier work [10,11]. In this paper, attention will exclusively be focussed on 3 layer feed-forward artificial neural networks. The symbol, $N_{i,j,k,l}$, as used in this paper will denote such a network with i inputs, j neurons in the first hidden layer, and l neurons in the last hidden (output) layer. The feed-forward nature of the network enables easy computation of the partial derivatives.

V. Neural Control System Design

Figure 4 shows the general implementation of the control system design. This system includes the reference model, the system under control (PM dc motor), and the dynamic neural network. The discrete model of the system under control is given in earlier work [10], and has the following form:

$$\omega(k+1) = \alpha\omega(k) + \beta\omega(k+1) + \gamma \mathrm{sign}(\omega(k)\omega^2(k)) + \delta \mathrm{sign}(\omega(k-1)\omega^2(k-1)) + \xi v_t(k) \qquad \textbf{(2)}$$

where v_t is the terminal voltage, α, β, γ, δ, and ξ are functions of the motor parameters, in addition to the sampling interval and the load torque. The controller input $v_t(k)$ at time step k can be approximated as a function $f(\cdot)$ of the shaft speed $\omega(k)$ in the form

$$v_t(k) = f(\omega(k+1), \omega(k), \omega(k-1)) \qquad \textbf{(3)}$$

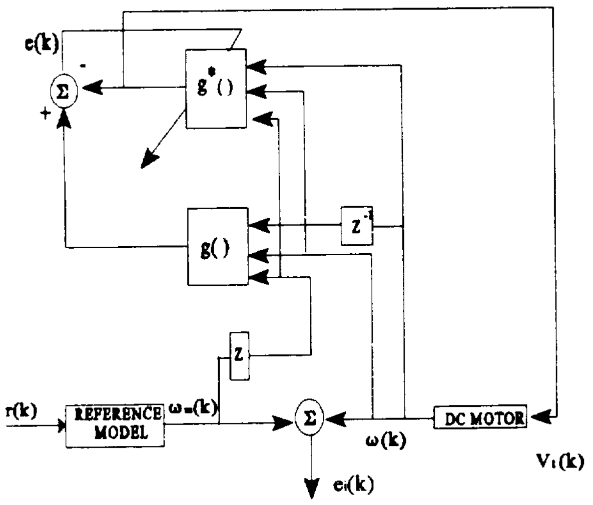

Figure 4. Neural Control System Design

A FANN is trained on-line to approximate $f(\cdot)$, as a function of its three input segment. In this case, the function approximated also equals the terminal voltage. The controller is designed to vary $v_t(k)$ in such a way that the speed $\omega(k)$ can follow a specified reference trajectory $\omega_m(k)$ for all k. Once the desired trajectory of the speed $\omega_m(k)$ is specified, the reference model generates the corresponding tracking residual r(k). The following reference model is selected .

$$\omega_m(k+1) = 0.6\,\omega_m(k) + 0.2\,\omega_m(k-1) + r(k) \qquad \textbf{(4)}$$

where r(k) is the bounded input to the reference model, and the constant coefficients are specifically chosen to make the unforced system asymptotically stable. Assuming the tracking error is small, the shaft speed at the (k+1)th instant can be predicted from;

$$\hat{\omega}(k+1) = 0.6\,\omega(k) + 0.2\,\omega(k-1) + r(k) \qquad \textbf{(5)}$$

This approximation to $\omega(k+1)$ is used as the input to the FANN along with $\omega(k)$ and $\omega(k-1)$. The network output $\hat{f}(\hat{\omega}(k+1), \omega(k), \omega(k-1))$, is computed and compared with $v_t(k)$, and the resulting error is used to train the FANN on-line. In the limit as $k \to \infty, \hat{f}(\hat{\omega}(k+1), \omega(k), \omega(k-1)) \to v_t(k))$, the control input to the dc motor, that results in $\omega(k)$ approaching $\omega_m(k)$.

VI. SIMULATION

When the object model is built and the control system is applied, closed-loop system simulation can be performed. To evaluate the closed-loop performance, trajectory control under several speed tracks, are investigated. In this case, a sinusoidal reference track is considered.

$$\omega_m(k) = 10\ \sin\underline{2\pi k\Delta T} + 16\ \sin\underline{2\pi k\Delta T} \qquad \textbf{(6)}$$

The FANN is trained on-line. The results are given in figures 5 through 10, which all show snapshots of the learning process. Thus, these figures provide an indication of how well the network identifier succeeds in learning (emulating) the nonlinear dynamics of the PM dc motor. It is clear from figures 5 through 10 that the tracking accuracy improves gradually with time. It is believed that improved performance can be obtained by further training of the network.

VII. CONCLUSIONS

The object-oriented control system design tool is an effective computer environment for applying and developing neural control schemes to motor drives. It integrates modelling , learning, design and simulation, and graphics user interface in one package. Motor modelling, parameter adjustments, and controller selection, can therefore be performed on-line. This design tool is an open-ended environment which can easily be extended due to the use of object-oriented programming and a knowledge base.

Object-oriented design technique can be used to develop a software model for the rocket engine numerical simulator executive. The model will be iterative and incremental. This will involve successive refinement of an object-oriented structure over each release to the next iteration of analysis and design. It is incremental in the sense that each pass of rocket engine numerical simulator and analysis and design cycle will lead to gradual refinement.

ACKNOWLEDGEMENT

This work is supported by NASA Lewis through grant number NAG3 - 1661. Mr. Donald F. Noga. Technical Officer.

REFERENCES

[1] G. Booch, Object Oriented Design with Applications, The Benjamin / Cummings Publishing ompany, Inc., 1991.

[2] B. Meyer, Object-Oriented Software Construction,Prentice Hall International, 1988.

[3] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, Object-Oriented Modeling and Design, Prentice Hall, 1991.

[4] F. L. Alvarado, R. H. Lasseter and Y. Liu, "An Integrated Engineering Simulation Environment", IEEE Transactions on Power Systems, Vol. 3, pp. 245-253, February 1988.

[5] S. Liu and S. M. Shahidehpour, " An Object-Oriented Power System Graphics Package for personal Computer environment", IEEE Transactions on Power Systems, Vol. 8, pp. 11054-1060, August 1993.

[6] M. Foley, A. Bose, W. Mitchell and A. Faustini, "An Object Based Graphical user Interface for Power Systems", Vol. 8, pp. 97-104, February 1993.

[7] J. Britton, "An Open, Object-Based Model as the Basis of an Architecture for Distributed Control Centers", IEEE Transactions on Power Systems, Vol. 2, pp. 1500-1508, Nov. 1992.

[8] M. Foley and Bose, "Object-Oriented On-Line Network Analysis, IEEE Transactions on Power Systems, Vol. 10, pp. 125-129, February 1995.

[9] K.S. Narendra and K. Parthasarathy, "Gradient -Methods for the Optimization of Dynamical Systems Containing Neural Networks," IEEE Trans. Neural Networks, Vol. 2, No. 2, pp. 252-262, March 1991.

[10] Ahmed Rubaai and R. Kotaru , "On-Line Identification and Control of a DC Motor using Dynamic Back-Propagation Neural Networks", Presented at the 1995 IAS Annual Meetings, paper No. TE95-20, Oct. 8-12, Orlando, Florida.

[11] Ahmed Rubaai and M.D. Kankam," Adaptive Real- Time Tracking Controller for High Performance Induction Motor Drives Using Neural Designs, " 1996 IEEE/IAS Annual Meeting,Vol. 3, pp.1709-1717, San Diego, California, Oct. 6-10, 1996.

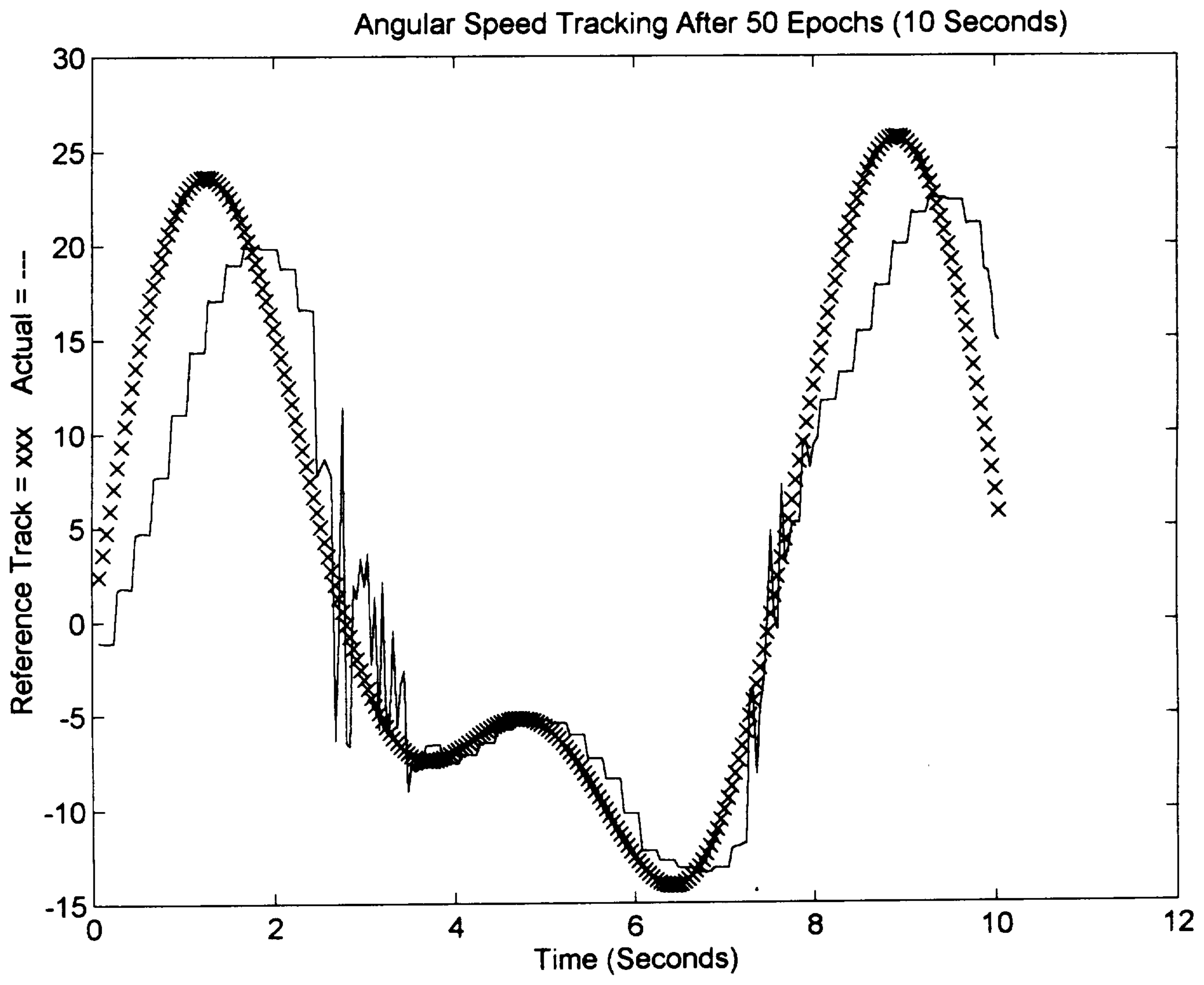

Fig. 5 Speed Control of a Sinusoidal Reference Track (10 Secs.)

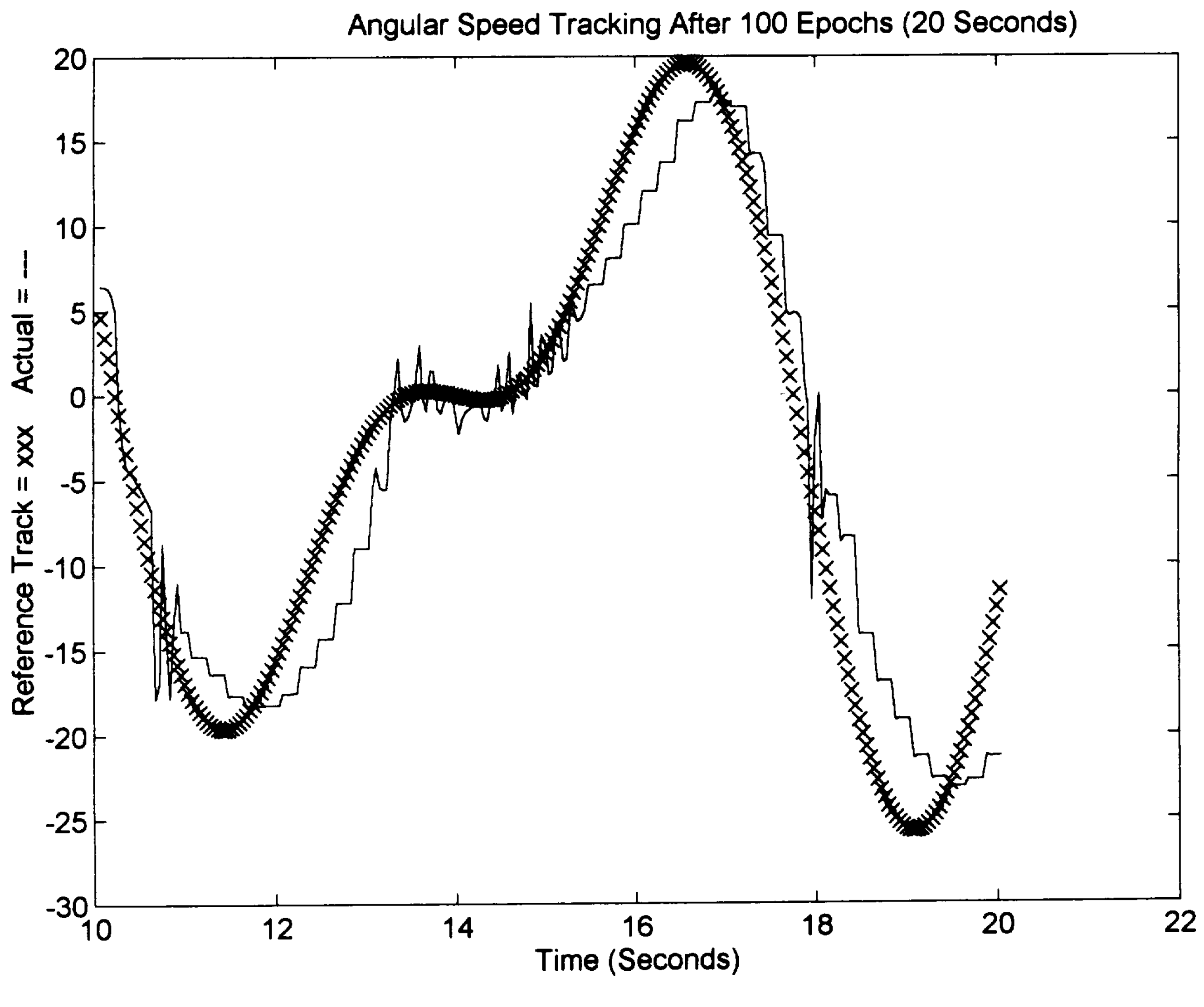

Fig. 6 Speed Control of a Sinusoidal Reference Track (20Secs.)

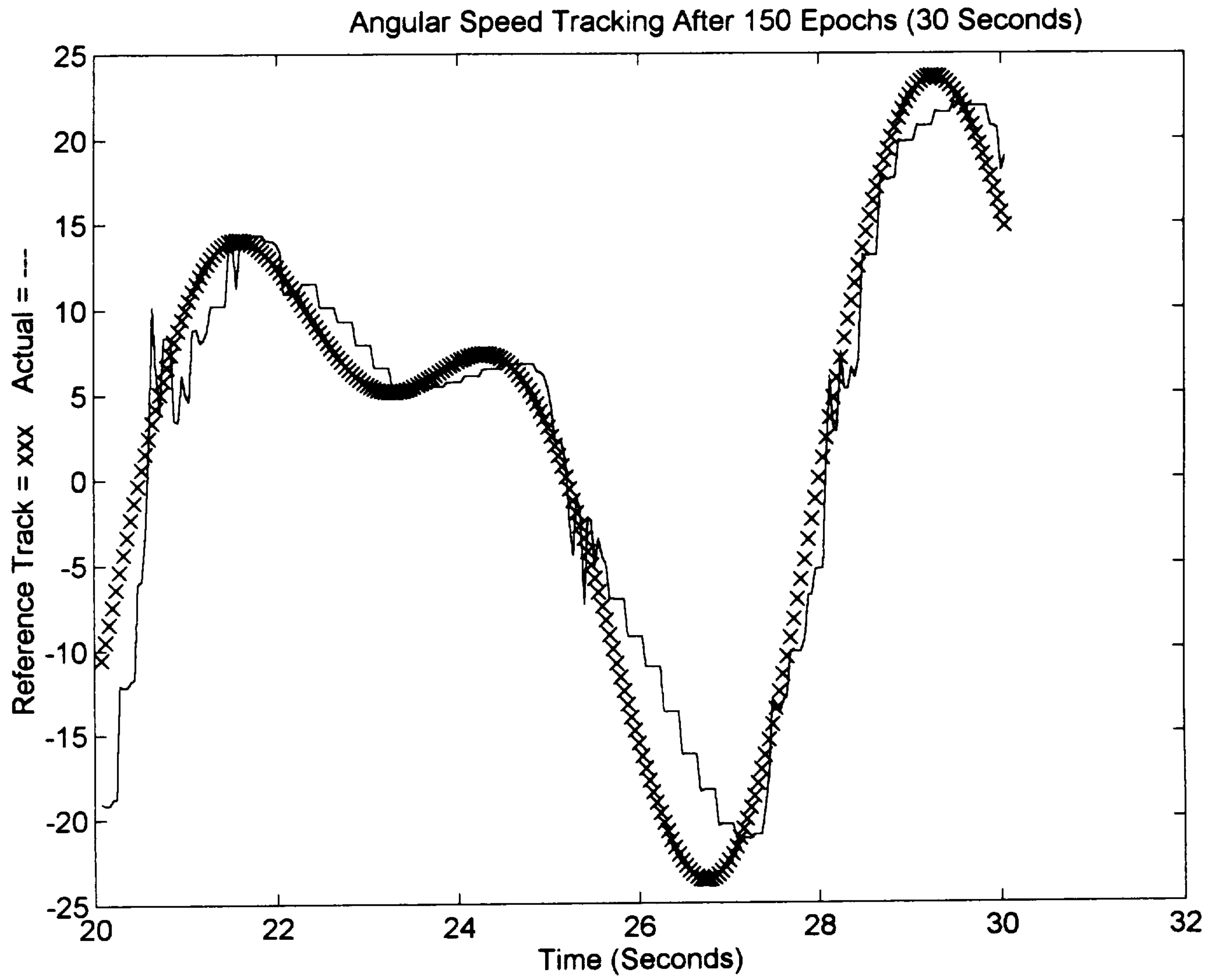

Fig. 7 Speed Control of a Sinusoidal Reference Track (30 Secs.)

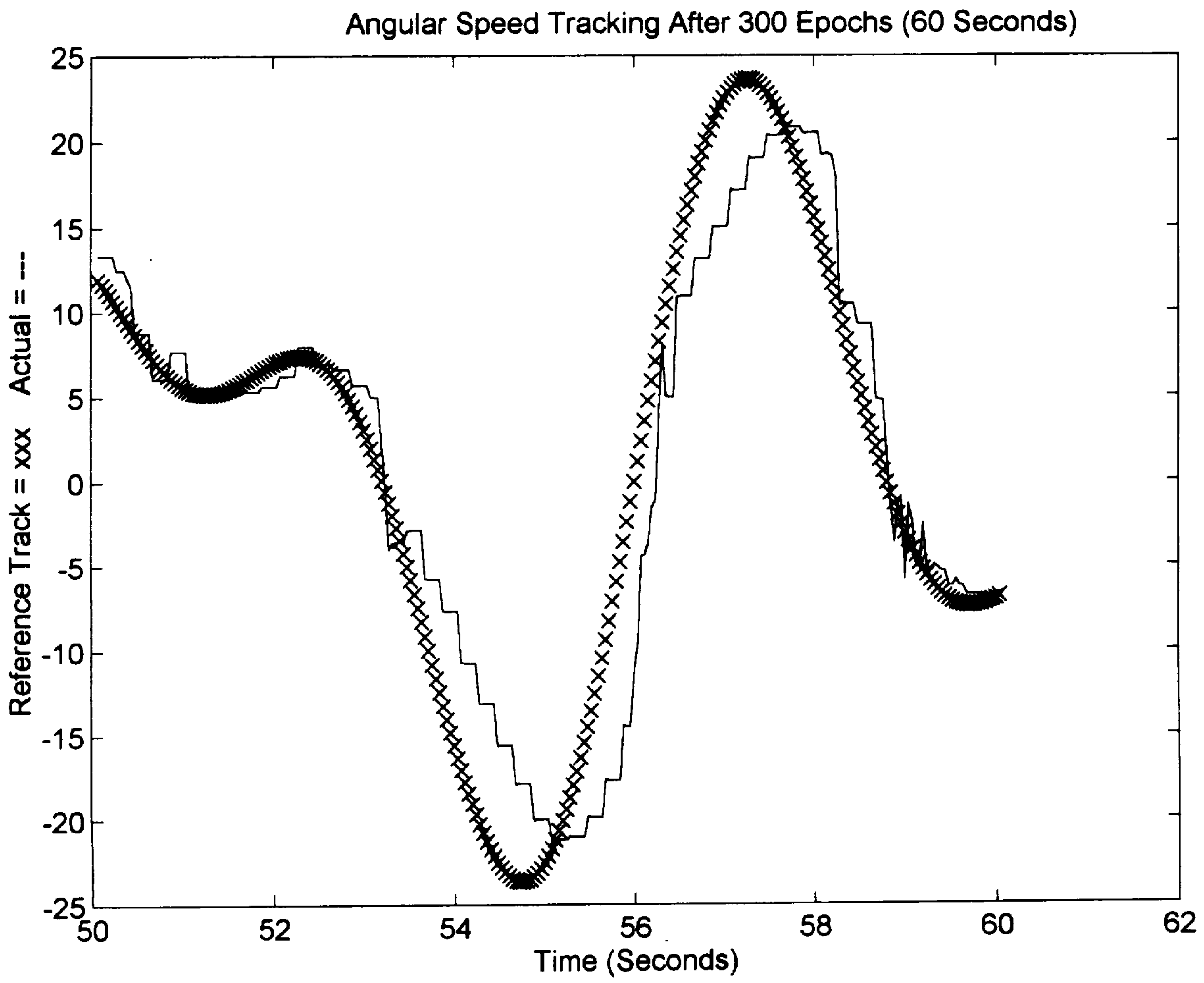

Fig. 8 Speed Control of a Sinusoidal Reference Track (60 Secs.)

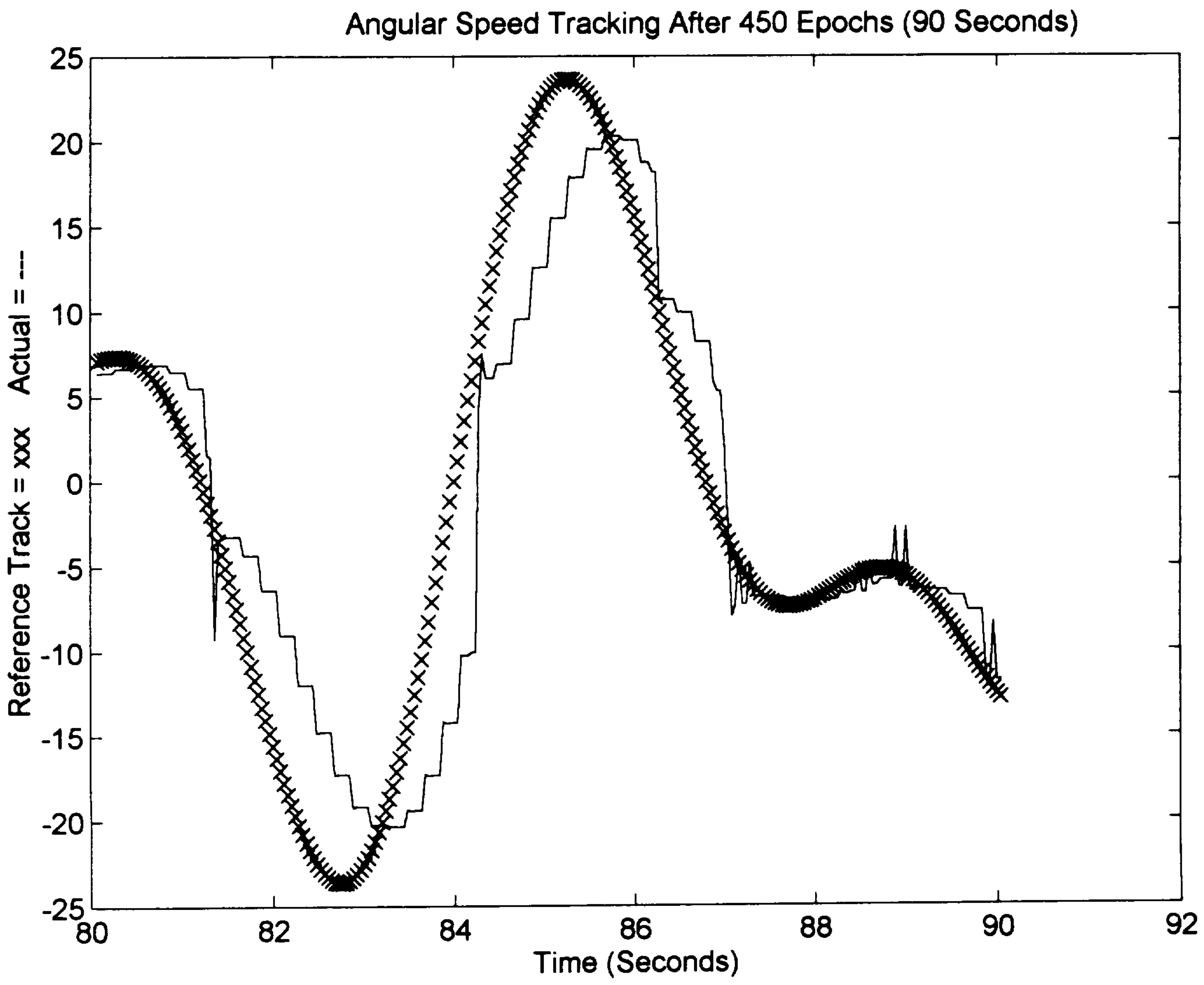

Fig. 9 Speed Control of a Sinusoidal Reference Track (90 Secs.)

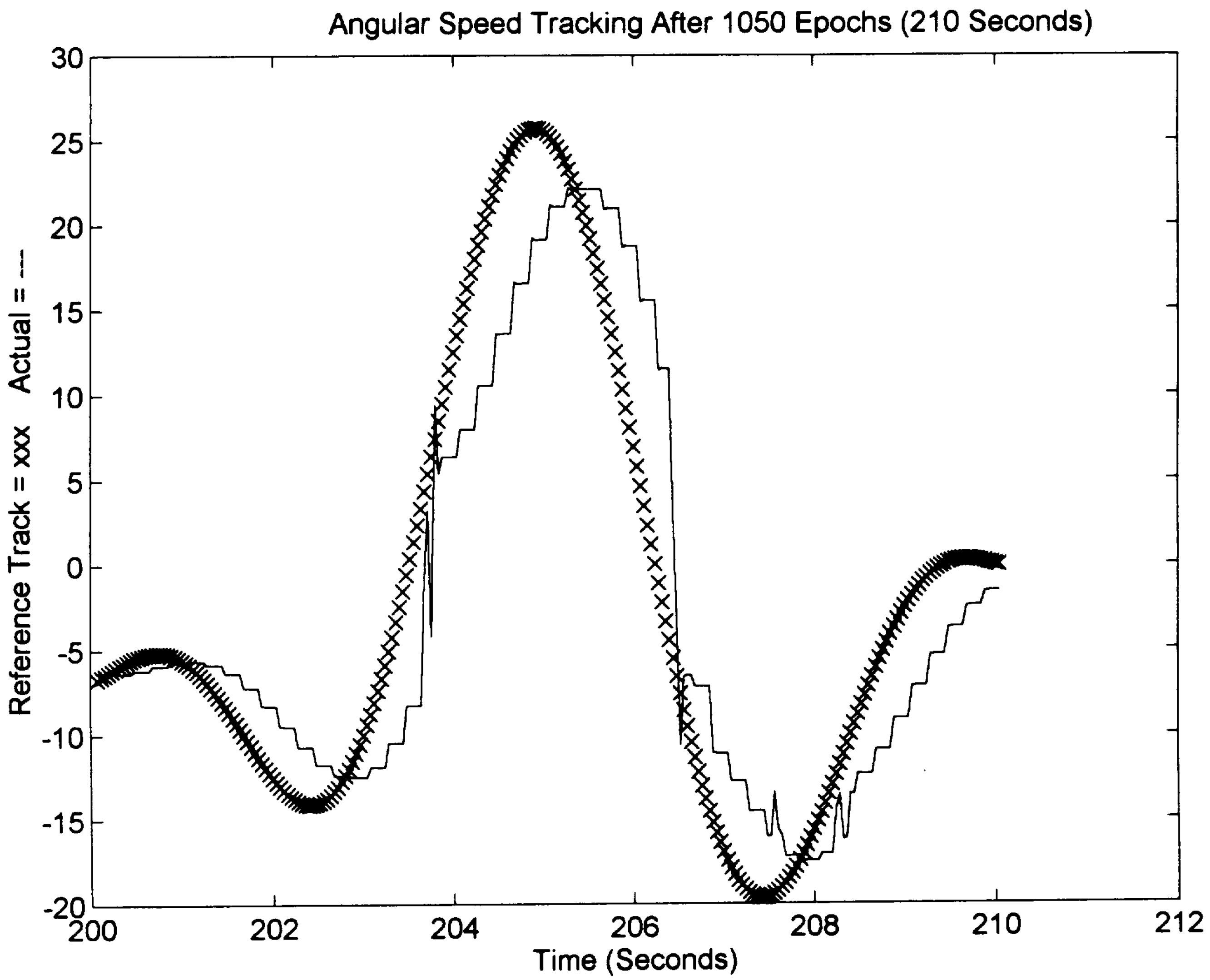

Fig. 10 Speed Control of a Sinusoidal Reference Track (210 Secs.)